Visualization of Unsteady Computational Fluid Dynamics

Final Techincal Report
for
Grant # NAG2-884

Submittted

by

Robert Haimes

Computational Aerospace Sciences Laboratory
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139

October 1995

Introduction

The current compute environment that most researchers are using for the calculation of 3D unsteady Computational Fluid Dynamic (CFD) results is a super-computer class machine. The Massively Parallel Processors (MPPs) such as the 160 node IBM SP2 at NAS and clusters of workstations acting as a single MPP (like NAS's SGI Power-Challenge array) provide the required computation bandwidth for CFD calculations of transient problems.

If we follow the traditional computational analysis steps for CFD (and we wish to construct an interactive visualizer) we need to be aware of the following:

- Disk space requirements
  A single snap-shot must contain at least the values (primitive variables) stored at the appropriate locations within the mesh. For most simple 3D Euler solvers that means 5 floating point words. Navier-Stokes solutions with turbulence models may contain 7 state-variables. The number can increase with the modeling of multi-phase flows, chemistry and/or electo-magnetic systems. If we examine a 5 equation system with 1 million nodes (with the field variables stored at the nodes) a single snap-shot will require 20 Megabytes. If 1000 time-steps are needed for the simulation (and the grid is not moving), 20 Gigabytes are required to record the entire simulation. This means that the workstation performing the visualization of this simulation requires vast amounts of disk space.

- Disk speed vs. Computational speeds
  The time required to read the complete solution of a saved time frame from disk is now longer than the compute time for a set number of iterations from an explicit solver. Depending on the hardware and solver an iteration of an implicit code may also take less time than reading the solution from disk. If one examines the performance improvements in the last decade or two, it is easy to see that depending on disk performance (vs. CPU improvement) may not be the best method for enhancing interactivity. Workstation performance continues to double every 18 months. The performance of commodity drives has gone from about 1 Megabyte/sec in 1985 to about 5 Megabytes/sec in 1995.

- Cluster and Parallel Machine I/O problems
  Disk access time is much worse within current parallel machines and cluster of workstations that are acting in concert to solve a single problem. In this case we are not trying to read the volume of data, but are running the solver and the solver outputs the solution. I/O is the bottleneck for a parallel machine with a front-end. The machine probably has the ability to compute in the GigaFLOP range but all this data has to be funneled to a single machine and put on disk by that machine. Clusters of workstations usually depend upon distributed file systems. In this case the disk access time is usually not the bottleneck, but the network becomes the pacing hardware. An IBM SP2 is a prime example of the difficulties of writing the solution out every iteration. The machine has a high-speed interconnect, but it is not currently used by the distributed file system. There are other access points into each node. Most SP2s have an Ethernet port for every node, some also have FDDI connections. These traditional network interfaces must be used for the file system.

- Numerics of particle traces
  Most visualization tools can work upon a single snap shot of the data but some visualization tools for transient problems require dealing with time. One such tool is the integration of

particle paths through a changing vector field. After a careful numerical stability and accuracy analysis of integration schemes (funded by previous NAS contracts) it has been shown that there exist certain time-step limitations to insure that the path calculated is correct. Even for higher order integration methods, the limitation is on the order of the time step used for the CFD calculation. This is because of a physical limit, the time-scale of the flow. What this means (for the visualization system) is that in order to get accurate particle traces, the velocity field must be examined close to every time step the solver takes.

Because of the disk space requirements and the time to write the solution to disk, the authors of unsteady flow solvers perform some sort of sub-sampling. This sub-sampling can either be spatial or temporal. Because the traditional approach is to deal with the data as if it were many steady-state solutions, this sub-sampling I/O is almost always temporal. The individual running the simulation figures the frequency to write the complete solution based on the available disk space. In many cases, important transitions are missed. Also since the solution is coarsely sampled in time, streaklines (unsteady particle paths as discussed above) almost always produces erroneous results. The problem with sub-sampling is that the time-step selected for the visualization becomes based on the available disk space and not the physical problem.

pV3 Status

Work is in progress on a set of software tools designed specifically to address visualizing 3D unsteady CFD results in these super-computer-like environments. The above issues are resolved by co-processing the visualization. The visualization is concurrently executed with the CFD solver. The parallel version of **Visual3, pV3** required splitting up the unsteady visualization task to allow execution across a network of workstation(s) and compute servers. In this computing model, the network is almost always the bottleneck so much of the effort involved techniques to reduce the size of the data transferred between machines.

The following design goals for **pV3** have been met:

- High Performance
  Take advantage of the proper hardware to get the best performance out of the entire compute arena. **pV3** requires graphics hardware so that scene rendering time is not a limitation and the data presented to the investigator is of high quality and timely. Also, most visualization techniques are embarrassingly parallel (based on elements within the computational volume). The execution of these tools is done within the partitioning performed to parallelize the CFD solver.
- Interactive
  The goal of any scientific visualization package should be to allow the assimilation of the vast amounts of data produced by the models and solvers in order to better understand the underlying physics. The ultimate goal, with this new knowledge, is to affect design and produce a better car, aircraft, gas-turbine engine, etc. This can only be done by interactively poking and probing into the data to interrogate areas of interest.
- Co-processing
  An important part of **pV3** is the ability to visualize the data as the solver or model progresses in time. It is also designed to allow the solver to run as independently as possible. If the solution procedure takes hours to days, **pV3** can *plug-into* the calculation, allow viewing of the data as it changes, then can *unplug* with the worst side-effect being the temporary allocation of memory and a possible load imbalance.

- **Visual3** functionality and programming
  pV3 provides the same kind of functionality as **Visual3** with the same suite of tools and probes. The data represented to the investigator (the 3D, 2D and 1D windows with cursor mapping) is the same. Also the same Graphical User Interface (GUI) is used.

  For the desired flexibility and the merging of the visualization with the solver, some programming is required. The coding is simple; like **Visual3**, all that is required of the programmer is the knowledge of the data. Learning the details of the underlying graphics, data extraction, and movement (for the visualization) is not needed. If the data is distributed in a cluster of machines, **pV3** deals with this, resulting in few complications to the user.

pV3 Rev 1.05 was released in April. It is anticipated that Rev 1.10 will be released before the end of the current contract. This port will include support the 'Batch' system described in the Status Section. The following machines are (and will be) supported as 'clients' (the computers containing the volume of data and performing the solver):

- CONVEX
- DEC Alphas running OSF/1
- DEC Stations (MIPS) running ULTRIX
- HP 9000/700 series at HP-UX 9.0 (or higher)
- IBM RS/6000s including the SP1s and SP2s
- SGI 4D Series, PI, Indigo, Indy, Power Series, Crimson, Onyx or Challenge running IRIX 5.x
- SGI PowerIndigo, PowerOnyx or PowerChallenge (all R8000 processors) running IRIX 6.x
- SUN running SunOS or Solaris

At Rev 1.05, the only machines supported as the **pV3** server were SGI workstations with 3D graphics support. Rev 1.10 will also include a server for IBM RS/6000s (at AIX 4.1) with 3D graphics adapters and OpenGL.

Presentations

An Analysis of 3-D Particle Path Integration Algorithms (with D. Darmofal), AIAA 95-1713
AIAA Computational Fluid Dynamics Conference, San Diego CA, June 1995.

Identification of Swirling Flow in 3-D Vector Fields (with D. Sujudi), AIAA 95-1715
AIAA Computational Fluid Dynamics Conference, San Diego CA, June 1995.

Concurrent Distributed Visualization and Solution Steering (Invited talk).
Parallel CFD '95, Pasadena CA, June 1995.

Unsteady Visualization of Grand Challenge Size CFD Problems: Traditional Post-Processing vs. Co-Processing.
ICASE/LaRC Symposium on Visualizing Time-Varying Data, Williamsburg, VA, Sept. 1995.

Demonstrations

AIAA Aerospace Sciences Meeting & Exhibit, Reno, January 1995.
ICASE/LaRC Symposium on Visualizing Time-Varying Data, Williamsburg, VA, Sept. 1995.
Supercomputing '95, San Diego, December 1995 (planned).

Status of Current Work

There were three major parts to the work performed within the period of this contract. One was the continued development of **pV3**. This also requires the second; addressing new and proposed standards for graphics and message passing software. And finally, continued work on feature extraction and identification for transient applications.

pV3 enhancements and development:

- Batch server
  The problem with **pV3** in a production environment or for batch execution is that the user may not be around to fire-up the server and view the results. An important part of this years' work was the design and implementation of a 'batch' server for **pV3**. The client side remains unchanged. The CFD solver need not know if the results are currently being viewed or to be viewed at some later time.

  Therefore, when a 'batch' job starts, the 'batch' **pV3** server is also started. Data is read on where and what tools and probes are to be active and their locations. The results (tool extracts) are collected and written to disk for play-back later. This is different from the normal post-processing in that the entire volume of data is not written to disk every iteration.

  The end result is something that is not interactive in the placement of tools, but can be thought of as analogous to a wind-tunnel experiment. You have to be smart in where you place probes to extract data of interest. If you miss an important area (or only find it after viewing these results) you will have to re-run the tunnel adding (or changing the location) of the probes. A post-processing viewer has also been developed to read and display the extracts. This viewer is highly interactive in dealing with time, in that the amount of data has been reduced by orders-of-magnitude.

- Visualization Extract Data System (VEDS)
  During the development of **elVis** at NASA Ames a new API for data extraction has been constructed. It was originally proposed to use fGL (the underlying disk structure that is handled by the API) for the basis of the **pV3** 'batch' system. Because of the constantly changing definition of fGL, it was not used as the disk representation of the co-processing extracts. The file format VEDS has been defined and is used for the disk resident objects. This format has been reviewed by NASA Ames, CEI (developers of EnSight), Intelligent Light (FieldView) and Untied Technologies. This file specification has been appended to this document.

New Software Standards:

- **MPI** - A Message Passing Interface
  pV3 currently uses **PVM** for all message passing a network related activities. A new message passing standard has emerged. The Message Passing Interface, **MPI**, is an attempt to deal with the MPP environment and shows much better efficiencies than **PVM** with the same level of message passing functionality. It has been shown (on NASA Ames' IBM SP2) that **PVM** and **MPI** can be used on the same task (**MPI** for the solver and **PVM** for the visualization). But because of the added complexity in using a mixed environment and the potential increase in performance, a pure **MPI** version of **pV3** was proposed.

For **MPI** to be used as the **pV3** low-level message passing protocol, the following points were addressed:
(1) Heterogeneous machines. The 'interactive' server currently requires a workstation off any MPP. The clients can be any machine, but specifically the IBM SP2 is of interest.
(2) Peer-to-peer. The **pV3** server must be started from the graphics workstation where the visualization session is to run. This may not be known *a priori.*
(3) Receive a message without knowing (before hand) that a message of this type will be sent.
(4) The ability to pull specific messages off a message queue and be able to select the next message based on a prioritized scheme.

After a complete assessment it was found that an **MPI** port is not possible. Point (1) above, is not handled. **MPI** on the SP2 does not support heterogeneous clusters, only nodes on the high-speed interconnect may be used. Also, **MPI** has no job control. The assumption is that all tasks that are running have started at the initial time. There is no mechanism to add the **pV3** server to an executing session, point (2) above. To reduce the complexity of a mixed **PVM** and **MPI** environment, **pV3**'s client-side API has been modified to remove all references to **PVM** IDs.

- OpenGL
  Two OpenGL ports of the **pV3** server were written during this contract. The first was to SGI equipment running IRIX 6. This also required that all code be '64-bit clean'. The second server port was to IBM RS/6000s. This port required AIX 4.1 for multi-threading and a machine with graphics hardware that supports OpenGL.

Feature Extraction and Identification:

In the past, feature extraction and identification were interesting concepts, but not required to understand the underlying physics of a steady flow field. This is because the results of the more traditional tools like iso-surfaces, cuts and streamlines were more interactive and easily abstracted so they could be represented to the investigator. These tools worked and properly conveyed the collected information at the expense of much user interaction. For unsteady flow-fields, the investigator does not have the luxury of spending much time scanning only one 'snap-shot' in the simulation. Automated assistance is required in pointing out areas of potential interest in the flow. This must not require a heavy compute burden (the visualization should not significantly slow down the solution procedure). And methods must be developed to abstract the feature and display it in a manner that makes sense physically.

Some success has been made within the scope of previous years work. A method that finds the core of vortices has been developed and was presented at the AIAA CFD conference this year. This is important for flow regimes that are vortex dominated (most of these are unsteady) such as flow over delta wings and flow in turbomachinery. Tracking the core can give insight into controlling unsteady lift and fluctuating loadings due to core/surface interactions.

It is hoped that a technique that finds regions of re-circulation will be complete within the time frame of this contract.

File Specification for Visualization Objects

The goal of this document is to specify a disk file structure, at the byte level, to be used by visualization systems in dealing with transient data. For these applications, the entire flow field is not written, but the results of the visualization extraction techniques are put on disk. The file spec must be flexible, since it is not known at this time all of the objects that need to be written. Therefore the file must be self-describing. Readers (viewers) that do not understand particular objects can ignore them (and optionally output a warning). This specification is being driven by the requirement to put pV3 extracts on disk, and therefore it will handle objects that have been spatially decomposed.

Primitives: There are basically 4 primitives used in this spec. IEEE 32-bit reals [R], IEEE 32-bit integers [I], 32 character titles [T] and 8 character markers are the primitives. Some records will have vectors of either R and/or I. The markers are only used to delimit entries in the file and not for data. All data is written as a binary stream using the non-DEC byte ordering (this includes HP, IBM, SUN and SGI). DEC and Intel machines must perform byte swapping during IO.

Notation: [X], [Xvar] or [countX] defines a (number of) primitive(s) within a record, where X = {R, I, T}. var - defines a variable (named var) of that type. The value assigned is taken from within the file and will be used later within the file definition. count - used to specify a number of primitives of the specified type within the record. The brackets are not part of the data stream and are only for ease of reading this document. count may be a constant, previously defined integer variable, or a simple predefined integer expression.

Multi-discipline: This file spec will handle multi-disciplinary cases. For this document, the definition of multi-discipline means that there are objects within the file that refer to data (either in the same volume, or from a different volumes/simulations) that have different field data (scalar, vector or state vectors).

Partitions: As previously mentioned, this spec supports and maintains the partitioning within a discipline. It is the responsibility of the reader to put together the pieces of an object for viewing.

Entities: An entity is a visualization object such as a geometric cut, iso-surface, streamline, etc. Entities may have many sub-entities, each defines a part of the entity (disjoint tris, disjoint quads, cell indices, field variables, etc). There may be as many sub-entities of each type as there are partitions. Sub-entities do not depend on the discipline.

Field Variables: As mentioned above, field variables are discipline dependent. These include scalars, vectors, tensors and state vectors. All field variables are associated with entities, and are treated like sub-entities.

Groups and Instances: There may be many entities of the same type (within the same discipline), in a single time frame, (i.e. 3 iso-surfaces of Mach Number at different values) each with a separate instance number. The instance number may be used to track an entity from one time frame to the next (the instance number should not change over time). Entities of a type may be grouped together (using a group index). For example, a line of streamlines will each be an entity, all having the same group number. This can be used for changing the viewing attributes of the entire group.

The file structure:

Markers	Comments
S*FILE**[Rspec_rev]	Start file
S*HEAD**	Start header
	Header records
E*HEAD**	End header
S*FRAME*[Rtime]	Data frame w/ time
	Data records
E*FRAME*[Rtime]	
S*FRAME*[Rtime]	Data frame w/ time
	Data records
E*FRAME*[Rtime]	
:	
:	
S*FRAME*[Rtime]	Data frame w/ time
	Data records
E*FRAME*[Rtime]	
E*FILE**	End file

As can be seen from this file skeleton, there is a header and then blocks of data. Each block contains all the information for the specified time. The header is used to predefine the types of information to be encountered within the time blocks and allows the data viewer to set-up titles and scaling for the data.

Blocks of data are delimited with start (S*) and end (E*) records. There is enough common information in both records that simple consistency checking is possible during reading. Only certain records are allowable within HEAD, FRAME and ENTIT blocks. These are described below.

The spec_rev field defines the revision level that this file conforms to.

Header records (order within the HEAD delimiters depends on record):

WRITER**[T][R]
Specifies the program that produced this file and its version number.

DATETIM*[T]
Specifies the start date and time that the file was created.

NDSCPLN*[Indisc]
Sets the number of disciplines to ndisc. If this record is not in the header, it is assumed that there is 1 discipline.

DSCPLNE*[T][Idisc][Inpart]
Specifies a discipline, gives it a name, an index and specifies the number of partitions. The index should run from 0 to ndisc - 1. If there are no records of this type, ndisc must be 1, the discipline will have no name, it has an index of 0, and there is 1 partition. The NDSCPLN* record must precede this entry (if the number of disciplines is greater than one).

PARTNUM*[T][Idisc][Ipart]
This defines a partition, gives it a name, associates it with a discipline and sets an index. The partition index should run from 0 to npart - 1. This record is not needed if there is only 1 partition. In this case the partition would not have a name and the partition index is 0. This record must come after any the DSCPLNE* description for this discipline.

SCALAR**[T][Idisc][Ifield][R][R] Field Item Specification
The SCALAR** record defines a title and field index for a scalar field variable within a particular discipline. The field index must be unique (positive and non-zero) for all field items defined within the discipline (scalars, vectors, tensors and state vectors). The 2 reals specify the default color range limits for startup. This record must come after any the DSCPLNE* description for this discipline. slen = 1.

VECTOR**[T][Idisc][Ifield][R] Field Item Specification
The VECTOR** record defines a title and field index for a vector field within a particular discipline. The field index must be unique for all field items defined within the discipline. The real specifies the default scaling for startup. This record must come after any the DSCPLNE* description for this discipline. slen = 3.

TENSOR**[T][Idisc][Ifield] Field Item Specification
The TENSOR** record defines a title and field index for a tensor field within a particular discipline. The field index must be unique for all data items defined within the discipline. This record must come after any the DSCPLNE* description for this discipline. It is assumed that all tensors are 3x3, therefore slen = 9.

STATE*V*[T][Idisc][Islen] Field Item Specification

The STATE*V* record defines a title with an implicit field index of zero for the disciplines state vector. The integer specifies the length of the state vector. This record must come after any the DSCPLNE* description for this discipline.

NOTE: when using state vectors the viewer needs to be given information on how to translate the state vector into scalar, vector and tensor field variables.

SURFACE*[T][Idisc][Isurfi]

This defines a bounding surface for a discipline. The title can be used by the viewer for selection and changing attributes. The index must be unique for each surface within the discipline.

ENTITY**[T][Ienti][Inint][Inreal][Insubs]

This record defines an entity (such as a geometric cut, iso-surface and etc). The entity is given a title and an index (that must be unique for all entities). The additional integers define the number of entity specific integer and real data items as well as the number of sub-entities.

NOTE: implementors need to standardize on entity definitions (based on title) and sub-entities, suggestions are described within this document.

SENTITY*[T][Ienti][Inint][Inreal][Insubs][Ifactor]

This record defines a special entity (such as a streamline) where the number of partition entries may be greater than the number of partitions. The entity is given a title and an index (that must be unique for all entities). The additional integers define the number of entity specific integer and real data items as well as the number of sub-entities. The factor limits the total number of segments to factor * npart. ENTITY** definitions have an implicit factor of 1.

NOTE: implementors need to standardize on entity definitions (based on title) and sub-entities, suggestions are described within this document.

DATADEF*[T][Ienti][Isubi][Iranki][Irankr]

This record specifies the title and index for a sub-entity associated with the entity index. The sub index must be between 0 and nsubs-1. ranki defines the number of integer entries for each count of the sub-entity. rankr defines the same for reals. Either ranki or rankr must be zero (all data entries are either integer or real). This record must come after the entity has been described via the ENTITY** or SENTIT* entry. For example, the definition of disjoint triangles based on indices would have ranki = 3 (3 nodes/tri) and rankr = 0.

Data records (within FRAME delimiters):

S*ENTIT*[Idisc][Ienti][Iinstance][T][Igroupi][nintI][nrealR]([factor*npartI])

The data for each entity is found between this record and the E*ENTIT* record. The only allowable items between these two markers are the SUBDATA*, FIELD***, PREVDAT*, and PREVFLD* records. The instance must be unique for each entity of the same type within a discipline and time-frame. The instance index should track across time. The title can be used by the viewer to give the object a name and should also track across time-frames. For Special Entities (fator not equal to 1) an additional field is required (factor*npart integers) that hold the actual partition indices for each segment.

E*ENTIT*[Idisc][Ienti][Iinstance]

This record marks the end of the data for this entity.

SUBDATA*[Ipart][Isubi][Ilen][len*rank Is or Rs]

This record defines the size and data of the sub-entity for this partition index. There should only be, at most, 1 record for each partition (except for special entities) and sub-entity index within the S*ENTIT* and E*ENTIT* delimiters. For entities declared via the SENTITY* tag, part may range from 0 to factor*npart-1 (for the specific discipline).

FIELD***[Ipart][Ifield][Ilen][len*slenR]

This record defines the size and field data associated with this partition index. field must be 0 for state vectors. Again, there should only be 1 record for each partition (except for special entities) and field index within the S*ENTIT* and E*ENTIT* delimiters

PREVDAT*[Ipart][Isubi][Ifp]

The PREVDAT* record indicates that the sub-entity data already exists in the file (it has been previously defined). fp points the the SUBDATA* record that contains the definition. fp is the byte index (absolute, from the beginning of the file) and can be used by fseek. This is very useful for reducing the overall file size, in that some sub-entities may not be changing in time. NOTE: For machines with 64 bit longs (DEC Alphas and SGI R8000s), the high order bytes must be zero filled (this limits the file size to 2^31 bytes).

PREVFLD*[Ipart][Ifield][Ifp]

This has the same effect as PREVDAT* except for field variables. fp points to the beginning of the proper FIELD*** record that contains the field data.

Suggested Sub-entities and their meaning

This first set can not have Field Items associated because the coordinates are part of the sub-entities:

[PolyLine] ranki: 0 rankr: 3
Defines a lines via the series of coordinate triads. The line length (len) is the number of points.

[Disjoint Lines] ranki: 0 rankr: 3
Defines a series of line segments from these coordinate triads. The number of segments is len/2.

[Disjoint Triangles] ranki: 0 rankr: 3
Defines a series of triangles from these coordinate triads. The number of triangles is len/3.

[Disjoint Quadrilaterals] ranki: 0 rankr: 3
Defines a series of quads from these coordinate triads. The number of quadrilaterals is len/4.

These sub-entities may be associated with Field Items. This association is done through the Coordinate sub-entity (which defines the node space to be referenced through).

[Coordinates] ranki: 0 rankr: 3
Define the number and location of the nodes that support the following sub-entities.

[Disjoint Tris] ranki: 3 rankr: 0
Each set of 3 integers index nodes within the Coordinates to specify a disjoint triangle.

[Disjoint Tris w/ Cell Index] ranki: 4 rankr: 0
The first 3 integers index a nodes within the Coordinates to specify a disjoint triangle. The fourth integer refers to the 3D cell that created the triangle.

[Disjoint Quads] ranki: 4 rankr: 0
Each set of 4 integers index nodes within the Coordinates to specify a disjoint quadrilateral.

[Disjoint Quads w/ Cell Index] ranki: 5 rankr: 0
The first 4 integers index a nodes within the Coordinates to specify a disjoint quadrilateral. The fifth integer refers to the 3D cell that created this facet.

[Mesh] ranki: 2 rankr: 0
This is a disjoint line definition where each of the two integers are indices into the Coordinates to define the line segments. These lines display the intersection of 3D cell faces and a cut surface (or bounding surface).

[Outline] ranki: 2 rankr: 0
This is a disjoint line definition where each of the two integers are indices into the Coordinates to define the line segments. These lines are the intersection of bounding surfaces and a cut surface. They are useful for a quick drawing mode.

[Outline w/ Surface Number] ranki: 3 rankr: 0
This is the same as Outline except for each line segment there is a tag to indicate the bounding surface index that was intersected.

The following sub-entities are associated with the node space and therefore must have the same length as Coordinates. This is also true for any Field data.

[Clip index] rakni: 1 rankr: 0
Used for planar cuts to indicate whether the node is viewable or not.

[2D Mapping] ranki: 0 rankr: 2
Define a mapping so the node (therefore surface) can be drawn in 2D.

[PseudoTime] ranki: 0 rankr: 1
Integration time for streaamlines. Note: the streamline is defined as a polyline from the Coordinates.

[Cell Index] ranki: 1 rankr: 0
Defines the 3D cell index that contains the point in Coordinates.

[Divergence] ranki: 0 rankr: 1
The cross-flow divergence experienced by the streamline (or particle). This allow streamtubes to be drawn.

[Angle] ranki: 0 rankr: 1
The angle of a streamribbon at the streamline position.

[Start Time] ranki: 0 rankr: 1
The seed time for the particle. Note: particles are individual points within Coordinates.

[Particle Number] ranki: 1 rankr: 0
The unique number (instance) for each particle. May be used to track a particle in time.

The following sub-entity exists as a catch all. The association with the entity and some coding is required in the viewer to use the data in this sub-entity.

[Generic Sub-Entity] rakni: ? rankr: ?

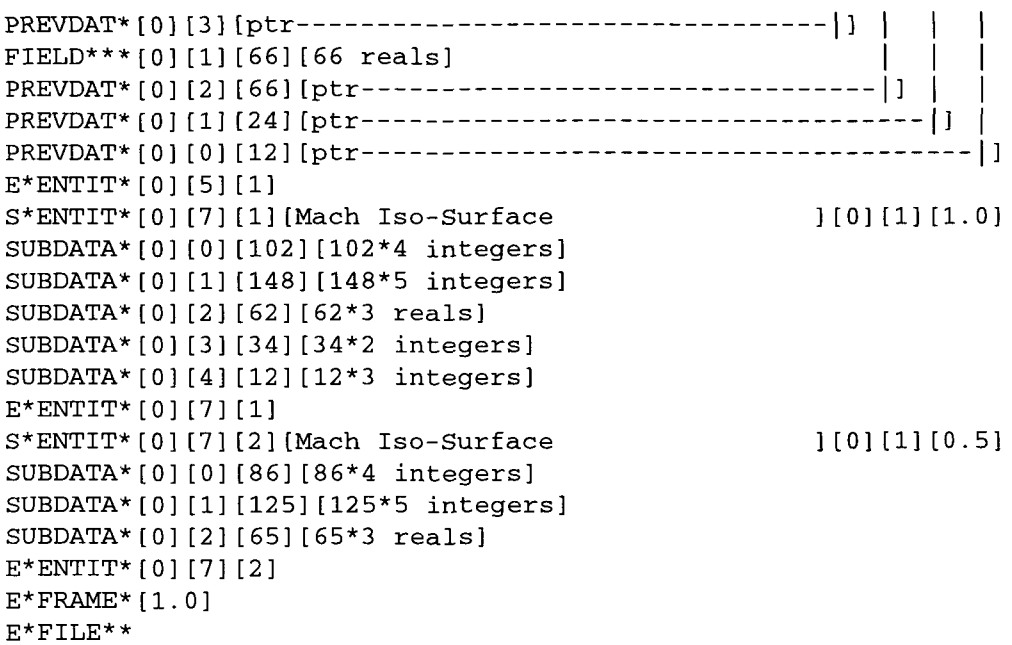
Entity and Sub-entities for some of the pV3 Extracts:

```
Title: Planar Cut              Entity Index: 2      nsubs: 6
       nints: 0                nreal: 9  (3 of the 4 corners in 3 space)
       0      Tris w/ Cell     ranki: 4             rankr: 0
       1      Quads w/ Cell    ranki: 5             rankr: 0
       2      Coordinates      ranki: 0             rankr: 3
       3      Mesh             ranki: 2             rankr: 0
       4      Outline w/ Surf  ranki: 3             rankr: 0
       5      Clip index       ranki: 1             rankr: 0

Title: Geometric Cut           Entity Index: 4      nsubs: 6
       nints: 1 (cut index)                        nreal: 1  (value)
       0      Tris w/ Cell     ranki: 4             rankr: 0
       1      Quads w/ Cell    ranki: 5             rankr: 0
       2      Coordinates      ranki: 0             rankr: 3
       3      Mesh             ranki: 2             rankr: 0
       4      Outline w/ Surf  ranki: 3             rankr: 0
       5      2D Mapping       ranki: 0             rankr: 2

Title: Domain Surface          Entity Index: 5      nsubs: 6
       nints: 1 (surface index)                    nreal: 0
       0      Tris w/ Cell     ranki: 4             rankr: 0
       1      Quads w/ Cell    ranki: 5             rankr: 0
       2      Coordinates      ranki: 0             rankr: 3
       3      Mesh             ranki: 2             rankr: 0
       4      Outline w/ Surf  ranki: 3             rankr: 0
       5      2D Mapping       ranki: 0             rankr: 2

Title: Iso-Surface             Entity Index: 7      nsubs: 5
       nints: 1  (field index)                     nreal: 1  (value)
       0      Tris w/ Cell     ranki: 4             rankr: 0
       1      Quads w/ Cell    ranki: 5             rankr: 0
       2      Coordinates      ranki: 0             rankr: 3
       3      Mesh             ranki: 2             rankr: 0
       4      Outline w/ Surf  ranki: 3             rankr: 0

Title: StreamLine              Entity Index: 18     nsubs: 5  (special)
       nints: 1  (surface/volume flag)             nreal: 3  (seed loc)
       0      Cell Index       ranki: 1             rankr: 0
       1      PseudoTime       ranki: 0             rankr: 1
       2      Coordinates      ranki: 0             rankr: 3
       3      Divergence       ranki: 0             rankr: 1
       4      Angle            ranki: 0             rankr: 1

Title: Particles               Entity Index: 19     nsubs: 4
       nints: 0                nreal: 0
       0      Particle Number  ranki: 1             rankr: 0
       1      Start Time       ranki: 0             rankr: 1
       2      Coordinates      ranki: 0             rankr: 3
       3      Divergence       ranki: 0             rankr: 1
```

A simple example:

In this example there is only one discipline and the results are from one partition.

```
S*FILE**[1.00]
S*HEAD**
WRITER**[pV3 Batch Server                ][1.20]
DATETIM*[ 6Jul95 12:48:00                ]
SURFACE*[Body                            ][0][1]
SCALAR**[Mach Number                     ][0][1][0.0][1.0]
STATE*V*[                                ][0][5]
ENTITY**[Domain Surface                  ][5][1][0][6]
DATADEF*[Disjoint Tris w/ Cell Index     ][5][0][4][0]
DATADEF*[Disjoint Quads w/ Cell Index    ][5][1][5][0]
DATADEF*[Coordinates                     ][5][2][0][3]
DATADEF*[Mesh                            ][5][3][2][0]
DATADEF*[Outline w/ Surface Number       ][5][4][3][0]
DATADEF*[2D Mapping                      ][5][5][0][2]
ENTITY**[Iso-Surface                     ][7][1][1][5]
DATADEF*[Disjoint Tris w/ Cell Index     ][7][0][4][0]
DATADEF*[Disjoint Quads w/ Cell Index    ][7][1][5][0]
DATADEF*[Coordinates                     ][7][2][0][3]
DATADEF*[Mesh                            ][7][3][2][0]
DATADEF*[Outline w/ Surface Number       ][7][4][3][0]
E*HEAD**
S*FRAME*[0.0]
S*ENTIT*[0][5][1][                                  ][0][1]
SUBDATA*[0][0][108][108*4 integers] <-----------------------|
SUBDATA*[0][1][138][138*5 integers] <---------------------|  |
SUBDATA*[0][2][66][66*3 reals] <----------------------|  |  |
FIELD***[0][1][66][66 reals]                          |  |  |
SUBDATA*[0][3][24][24*2 integers] <-----------------|  |  |  |
SUBDATA*[0][4][12][12*3 integers] <-------------|   |  |  |  |
E*ENTIT*[0][5][1]                               |   |  |  |  |
S*ENTIT*[0][7][1][Mach Iso-Surface                ][0][1][1.0]
SUBDATA*[0][0][100][100*4 integers]             |   |  |  |  |
SUBDATA*[0][1][150][150*5 integers]             |   |  |  |  |
SUBDATA*[0][2][64][64*3 reals]                  |   |  |  |  |
SUBDATA*[0][3][36][36*2 integers]               |   |  |  |  |
SUBDATA*[0][4][10][10*3 integers]               |   |  |  |  |
E*ENTIT*[0][7][1]                               |   |  |  |  |
S*ENTIT*[0][7][2][Mach Iso-Surface                ][0][1][0.5]
SUBDATA*[0][0][80][80*4 integers]               |   |  |  |  |
SUBDATA*[0][1][120][120*5 integers]             |   |  |  |  |
SUBDATA*[0][2][55][55*3 reals]                  |   |  |  |  |
E*ENTIT*[0][7][2]                               |   |  |  |  |
E*FRAME*[0.0]                                   |   |  |  |  |
S*FRAME*[1.0]                                   |   |  |  |  |
S*ENTIT*[0][5][1][                                ][0][1]    |
PREVDAT*[0][4][ptr------------------------------|] |  |  |  |
```


PREVDAT*[0][3][ptr
FIELD***[0][1][66][66 reals]
PREVDAT*[0][2][66][ptr
PREVDAT*[0][1][24][ptr
PREVDAT*[0][0][12][ptr
E*ENTIT*[0][5][1]
S*ENTIT*[0][7][1][Mach Iso-Surface][0][1][1.0]
SUBDATA*[0][0][102][102*4 integers]
SUBDATA*[0][1][148][148*5 integers]
SUBDATA*[0][2][62][62*3 reals]
SUBDATA*[0][3][34][34*2 integers]
SUBDATA*[0][4][12][12*3 integers]
E*ENTIT*[0][7][1]
S*ENTIT*[0][7][2][Mach Iso-Surface][0][1][0.5]
SUBDATA*[0][0][86][86*4 integers]
SUBDATA*[0][1][125][125*5 integers]
SUBDATA*[0][2][65][65*3 reals]
E*ENTIT*[0][7][2]
E*FRAME*[1.0]
E*FILE**